

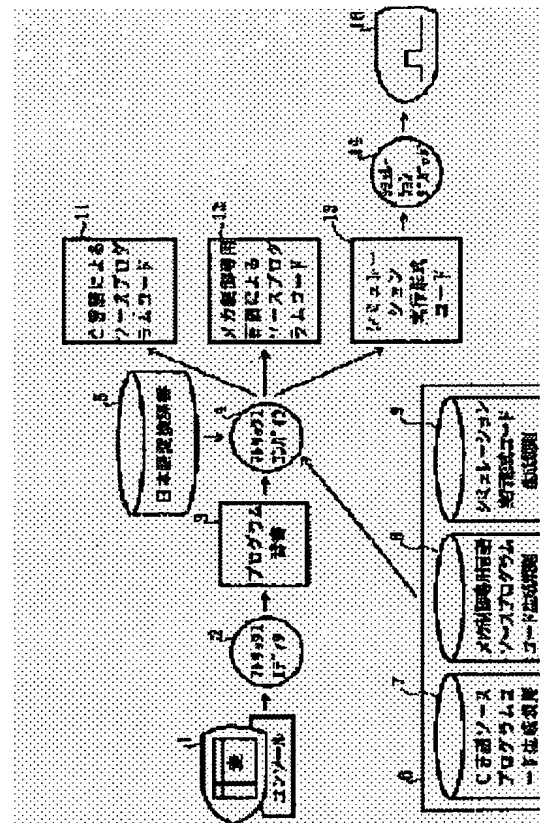
PROGRAM DEVELOPMENT SUPPORTING SYSTEM

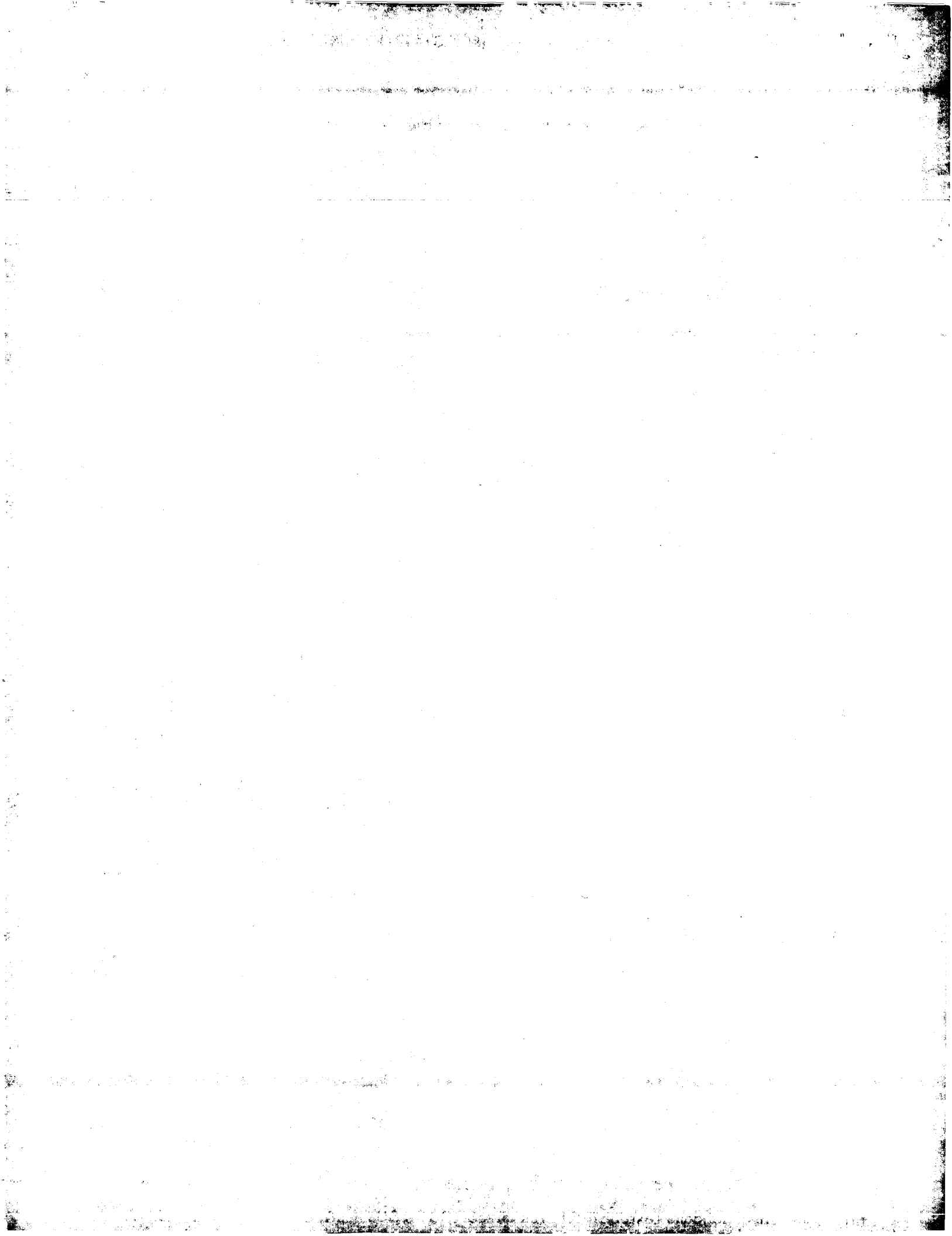
Patent number: JP9016382
Publication date: 1997-01-17
Inventor: OTSUKI JUNICHI; IKETANI TAKESHI
Applicant: OKI ELECTRIC IND CO LTD
Classification:
- international: G06F9/06
- european:
Application number: JP19950187894 19950630
Priority number(s):

Abstract of JP9016382

PURPOSE: To automatically provide a source program code by easily and simply describing a device integrated software.

CONSTITUTION: A program to be inputted by using a matrix editor 2 is expressed in a table format just like a tabulation software. When processing described in each column of a table visually described on a WS is successively executed from the head, the program can be executed. According to fixed rules, a matrix compiler 4 converts a program dictionary 3 corresponding to this table or the other definition drawing to the source program code in a C language, for example. Besides, the source program in a language dedicated to mechanism control is generated from the speciality of firmware as well.





(19) 日本国特許庁 (J P) (12) 公開特許公報 (A) 特開平9-16382
(43) 公開日 平成9年(1997)1月17日

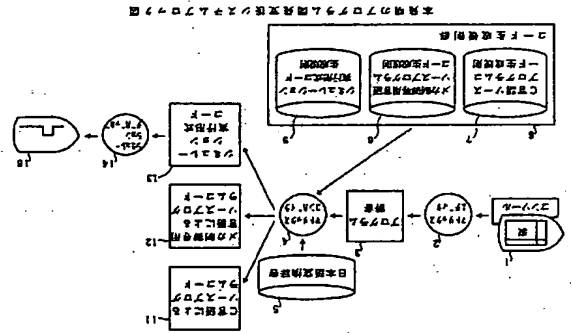
(51) Int. Cl. G06F 9/06	識別記号 530	F I G06F 9/06	530 G
審査請求 未請求 請求項の数 4 F D (全23頁)			
(21) 出願番号 特願平7-187894	(71) 出願人 000000295 沖電気工業株式会社 東京都港区虎ノ門1丁目7番12号	沖電気 工業株式会社内	
(22) 出願日 平成7年(1995)6月30日	(72) 発明者 大機 純一 東京都港区虎ノ門1丁目7番12号	沖電気 工業株式会社内	
	(72) 発明者 池谷 健 東京都港区虎ノ門1丁目7番12号	沖電気 工業株式会社内	
	(74) 代理人 井理士 佐藤 幸男 (外1名)		

(54) 【発明の名称】 プログラム開発支援システム

(57) 【要約】

【目的】 装置組み込みソフトウェアを容易に簡便に記述し、自動的にそのソースプログラムコードを得る。

【構成】 マトリクスエディタ2を用いて入力するプログラムは、丁度表計算ソフトのような表形式で表現される。WS上にビジュアルに記述される表の各欄に記入された処理を先頭から順に実行すればプログラムが実行できる。マトリクスコンパイラ4は、この表やその他の定義等に対応するプログラム辞書3を一定の規則に従って、例えばC言語によるソースプログラムコードに変換する。また、ファームウェアの特殊性からガガ制御専用言語によるソースプログラムコードの生成も行う。



【特許請求の範囲】

【請求項1】 プログラムを構成する各処理の内容を表の各欄に記入して、先頭から順にその表の内容を実行することにより、そのプログラムを実行できるように表示した表を入力するマトリクスエディタと、

このマトリクスエディタで入力され生成されたプログラム辞書に、ソースコード生成規則を適用して、前記表に対応するソースプログラムコードを生成するマトリクスコンパイラとを備えたことを特徴とするプログラム開発支援システム。

【請求項2】 プログラムを構成する各処理の内容を表の各欄に記入して、先頭から順にその表の内容を実行することにより、そのプログラムを実行できるように表示した表を入力するマトリクスエディタと、

このマトリクスエディタで入力され生成されたプログラム辞書に、シミュレータ実行形式コード生成規則を適用して、前記表に対応するシミュレータ実行形式コードを生成するマトリクスコンパイラとを備えたことを特徴とするプログラム開発支援システム。

【請求項3】 マトリクスエディタは、モジュール定義図とデータ形式定義図と構造体定義図とデータ意味定義図と表の入力を行い、C言語によるソースプログラムコード生成のための意味解釈用のプログラム辞書を作成することを特徴とする請求項1記載のプログラム開発支援システム。

【請求項4】 メカ動作規則とシミュレータ実行形式コードとを参照して、プログラムの各命令が実行されたときの制御対象装置の各部の状態を表す状態情報を作成する状態更新部と、

前記状態情報に従って状態図を生成してプログラムの任意のステータスでその状態図を表示する請求項2記載のプログラム開発支援システム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、プログラム開発、特に装置組み込みソフトウェアの開発支援に利用されるプログラム開発支援システムに関する。

【0002】

【従来の技術】 ソフトウェアの1分野に装置組み込みソフトと云われるものがある。これは、ファームウェアと呼ばれている。ファームウェアはハードウェアとアプリケーションソフトウェアの中間に位置し、メカ制御、通信制御、入出力制御等のハードウェア制御を主に担当する。このファームウェアは一般のソフトウェアと比べてリアルタイム性の強い要求があり、マルチCPU、マルチタスクといった並行処理を要求するといった特徴がある。従って、一般のソフトウェアがハードウェアの独立化標準化に伴ってパッケージ化の傾向にあるのに対し、ファームウェアは処理時間やハードウェア資源の強い制約の

ため、装置毎に最適な構成がとられ、標準化及び生産性向上が困難な特徴を持つ。

【0003】 従って、一般汎用性の高いソフトウェアについては、各種の簡便な開発ツールが登場し利用されているが、ファームウェアの開発支援のためには、例えばプログラムのフローチャートと解析してソースプログラムのコードを得るといったものが紹介されているに過ぎず(特公平6-40302号公報)、より具体的な簡便なものとは実用化されていない。

【0004】

【発明が解決しようとする課題】 ところで、上記のような従来のプログラム開発支援システムには次のような解決すべき課題があった。プログラム開発支援は、プログラマーに対して比較的簡単なプログラム内容の記述を要求して、これを自動的に変換処理してソースプログラムコード等を得ることを目的とする。従って、プログラムをどう表現して開発支援システムに入力するかという点が問題となる。従来、このようなプログラムの表現が専門的な詳細な知識を必要とし、多くの変数の面に対する条件を満たすバグの無いプログラムを作成するためには、高度に熟練を必要があった。

【0005】 また、ファームウェア開発の生産性を上げるためには、できるだけ開発済みのファームウェアを流用することが重要である。しかしながら、作成済みのソースコードプログラムは必ずしもその処理内容を正確に短時間に把握することが容易でないという問題があった。

【0006】

【課題を解決するための手段】 本発明は以上の点を解決するため次の構成を採用する。本発明のプログラム開発支援システムは、プログラムを構成する各処理の内容を表の各欄に記入して、先頭から順にその表の内容を実行することにより、そのプログラムを実行できるように表示した表を入力するマトリクスエディタと、このマトリクスエディタで入力され生成されたプログラム辞書に、ソースコード生成規則を適用して、表に対応するソースプログラムコードを生成するマトリクスコンパイラとを備える。

【0007】 本発明の別のプログラム開発支援システムは、プログラムを構成する各処理の内容を表の各欄に記入して、先頭から順にその表の内容を実行することにより、そのプログラムを実行できるように表示した表を入力するマトリクスエディタと、このマトリクスエディタで入力され生成されたプログラム辞書に、シミュレータ実行形式コード生成規則を適用して、表に対応するシミュレータ実行形式コードを生成するマトリクスコンパイラとを備える。

【0008】 なお、マトリクスエディタは、モジュール定義図とデータ形式定義図と構造体定義図とデータ意味定義図と表の入力を行い、C言語によるソースプログラ

ムコード生成のための意味解釈用のプログラム辞書を生
成することが好ましい。また、メカ動作規則とシミュレ
ータ実行形式コードとを参照して、プログラムの各命令
が実行されたときの制御対象装置の各部の状態を表す状
態情報生成した状態更新部と、状態情報に従って状態
図を生成してプログラムの任意のステップでその状態
を表示することが好ましい。

【0009】

【作用】マトリクスエディタを用いて入力するプログラ
ムは、丁度計算ソフトのような表形式で表現される。

WS (ワークステーション) 上にビジュアルに記述され
る表の各欄に記入された処理を先頭から順に実行すれば
プログラムが実行できる。マトリクスコンパイルは、こ
の表やその他の定義図等に対応するプログラム辞書を一
定の規則に従って、例えばC言語によるソースプログラ
ムコードに変換する。また、アームウェアの特殊性か
らメカ制御専用言語によるソースプログラムコードの生
成も行う。更に、プログラムのデバッグ作業を容易にす
るために、シミュレーション実行形式コードを生成す
る。例えば、1ステップずつシミュレーションを実行さ
れると、プログラムの制御対象となる装置各部の状態
をアニメーション化して表示する。これにより、分か
りやすく漏れが無くバグの少ないプログラム開発が可能
となる。

【0010】

【実施例】以下、本発明を図の実施例を用いて詳細に説
明する。

(システム構成) 図1は、本発明のプログラム開発支援
システムの具体例を示すブロック図である。図のシステ
ムは、コンソール1と、これを用いてプログラムを表現
するための表を入力するマトリクスエディタ2と、これ
に生成されたプログラム辞書3、マトリクスコンパイル
4、日本語変換辞書5、コード生成規則群6、ソースプ
ログラムコンパイルにより生成されたC言語によるソースプ
ログラムコード11、メカ制御専用言語によるソースプ
ログラムコード12、シミュレーション実行形式コード1
3、デバッグ作業のためのシミュレーションデバッグ1
4等から構成される。なお、図に示すディスプレイ16
はコンソール1に設けられたものを説明の都合上、右側
にも示すものでもある。

【0011】なお、このシステムは、例えば図2に示し
たようなハードウェアにより実現する。図2は、本発明
の実施のためのハードウェア説明図である。図の左側に
は、よく知られたワークステーション等のディスプレイ
21、制御部本体22及びキーボード23が斜視図で表
示されている。なお、キーボード23の他に、この装置
を操作するためにマウス26も利用される。また、この
装置の制御部本体22の内部には、図に示すように、中
央処理装置 (CPU) 24やハードディスク装置 (HD
D) 25等が設けられている。

【0012】図1に示すコンソール1は、図2に示した
ワークステーション本体を示す。また、マトリクスエデ
ィタ2やマトリクスコンパイル4、シミュレーションデ
バッグ14等は、ハードディスク装置25の中に格納さ
れた処理用プログラムによって構成される。プログラマ
辞書3、日本語変換辞書5、コード生成規則群6や、マ
トリクスコンパイル4により生成されたソースプログラ
ムコード11、12、シミュレーション実行形式コード
13等は、ハードディスク装置25やその他の記憶装置
に格納される。

【0013】このシステムは、開発支援の対象となるプ
ログラムを表示する定義図や表のような各種チャートの
編集機能と、プログラムソースコードの生成機能及びシ
ミュレーションデバッグ機能を持つ。各種チャートの編
集は、マトリクスエディタ2により行う。プログラムソ
ースコードの生成は、マトリクスコンパイル4により行
う。また、シミュレーションデバッグはシミュレーショ
ンデバッグ14により行う。

【0014】(マトリクスエディタの機能) 本発明で
は、アームウェアを如何に分かりやすく見せるかをフ
ームウェア生産性向上のキーファクタとして捉え、フ
ームウェアにマツチした表現形式として表を使用し
た。表はテキストスタイルの表現でありながら、総的で
分かりやすく自然な表現である。表により、バグが少な
い。マトリクスエディタ2は、この目的のために、プロ
グラマーの操作によるモジュール定義図、データ形式定
義図、構造体型定義図、データ意味定義図、表等の作
成の入力を受け付け、対応するプログラム辞書を生
成する。このシステムでは、プログラミンを、プログラ
ム辞書とは、プログラマーがマトリクスエディタ2に
よって作るモジュール定義、データ定義や表の内容を記
述した情報である。

【0015】以下、プログラムをC言語で記述するため
に入力すべき表の具体例を説明する。

(表の具体例) 図3～図9は、マトリクスエディタで入
力される表の具体例を示す。図3は、モジュール定義図
の説明図、図4は、データ形式定義図の説明図、図5
は、構造体型定義図の説明図、図6は、データ意味定義
図の説明図、図7は表の例説明図である。なお、本発明
においては、上記モジュール定義図、データ形式定義
図、構造体型定義図、データ意味定義図及びその後で説
明するマトリクス表の表を含めて単に表というように表
現している。即ち、これらの表がマトリクスエディタに
より入力されることによって、最終的に図1に示すマ
トリクスコンパイル4が動作して必要なソースプログラ
ムコード等が生成される。

【0016】図3に示すモジュール定義図は、これから
50 開発しようとするプログラムモジュール毎の入力31

と、これを処理するためのモジュール32と、出力33
との概要を明らかにしたものである。即ち、例えば図に
示すような入力変数A、入力変数B等を使用してモジュ
ールで一定の演算処理を実行し、その結果を出力する。
この場合の入力変数の概要やモジュールの処理概要、出
力されたデータの概要を説明しておくことによって、各
モジュールの役割をプログラマー本人や第三者が容易に
理解できるようにする。

【0017】図4に示すデータ形式定義図では、ファイ
ルポインタ34、変数35、36、配列37、ポインタ10
38等が定義され表形式に表現されている。これによ
り、データの形式やファイル形式を明確に表現する。こ
の例ではファイルポインタ34に「ファイルポインタ
A」が設定され、変数35には整数の「変数A」が設定
されるといった要領で定義が行われている。図5に示す
構造体型定義図によれば、構造体型名40や、メンバ名
41、メンバの型42、変数43、ポインタ44等が表
形式で明らかにされている。この例では、構造体型名4
0には「構造体型A」が設定され、以下、同様に変数等
が設定されている。

【0018】図3に示すモジュール定義図の入力31で
示した入力変数は、図4に示すデータ形式定義図等によ
って詳細に定義される。また、モジュール32の機能
は、図5に示した構造体型定義図等によって具体的に定
義される。そして、その処理の結果として得られる出力
33は、図6に示すようなデータ意味定義図によって明
確化される。即ち、図6の出力値の概要によれば、セン
サの出力45の意味がその下に具体的に表現されてい
る。即ち、出力の値が「1」であればセンサがONの状
態であり、出力値が「0」の場合にはセンサがOFFの状
態である。

【0019】図7に示した表は、モジュールによる具体
的な処理内容を表形式に記述したものである。例えば、
この表50は図に示すように、T=1から10までの範
囲でTをインクリメントしながら、処理51と処理52
を繰り返すといった内容になっている。処理51の内容
は、ある演算処理ABCとする。また、処理52の内容
は、パラメータX、Yの内容に応じて4種類に分けられ
ている。このような表形式のプログラム表記は、例えば
次のような基本形を組み合わせて行う。図7に示す処理
の内容は次の基本形の説明によって明らかにする。

【0020】図8には、基本形T1と基本形T2とを示
した。図8(a)は、基本形T1の内容を示し、左側が
条件記述欄、右側が処理内容の記述欄を示している。即
ち、図の例では、Xが1のときと2のときで処理内容を
分けている。Xが1のときは処理ABCを実行し、Xが
2のときは処理DEFを実行する。

【0021】(b)は基本形T2の内容を示し、XとY
をパラメータとしてそれぞれその組合せによって処理の
内容を切り替えている。即ち、Xが1でYが1の場合に50

は、処理ABCを実行し、Xが1でYが2の場合には、
処理DEFを実行する。また、Xが2でYが1の場合には
は、処理GHIを実行し、Xが2でYが2のとき、処理
JKLを実行する。(a)に示す例の場合、ソースコー
ドで記述しても表現上大きな差はないが、(b)のよう
な内容になると、ソースコードで記述した場合とこのよ
うな表形式で記述した場合とでは、まず一見して理解し
易いかどうかに大きな差がある。しかも、表形式の場
合、全ての条件を網羅し易く、記入漏れや記述ミスが生
じにくい。

【0022】図9には、基本形T3、T4、T5の説明
図を示す。(a)に示す基本形T3では、表を用いて繰
り返し演算を表現している。即ち、T=1から10まで
の範囲でTをインクリメントし、処理ABCと処理DEF
とを実行するといった内容になっている。特に、この
ような表の中に更に表を埋め込むネスト表現を実行する
場合、ソースプログラムコードでは、繰返し命令、繰返
し演算処理の仕切り、その他が入り乱れて、解釈も容易
でなく煩雑になる。しかしながら、このような表形式で
表現する場合には、図のABCという処理の中にこの図
と同様の繰返し表現を含めればよいだけで、どの繰返
し表現がどの繰返し表現の中に含まれるかあるいはどの繰
返し表現が何回繰返されるか等、極めて容易にビジュアル
に認識し区別できるといった効果がある。

【0023】(b)に示す基本形T4は、条件がアンド
やオアで組み合わされているとき、その条件に合えばい
くつかの処理を合わせて実行するといった表現を例示し
ている。即ち、ここでは、Xが「1」でYが「1」の両
方の条件が成立した場合に、処理ABCと処理DEFと
を順に実行する。なお、図(c)に示す基本形T5は、
これまでそれぞれ組合せとして説明した1個の欄に1個
の処理の内容を記述したもので、無条件に処理ABCを
実行するといった表現となっている。ほとんど全てのプ
ログラムは、上記のような表を組み合わせて連結すること
で表現できる。しかも、これにより分かり易く論理的な
ミスを生じにくいといった特徴を持つ。

【0024】(マトリクスコンパイルの機能) マトリク
スコンパイルは、マトリクスエディタ2が生成したプロ
グラム辞書から図1に示すようなC言語によるソースプ
ログラムコード11、メカ制御専用言語によるソースプ
ログラムコード12、シミュレーション実行形式コード
13を生成する部分である。

【0025】このマトリクスコンパイル4は、一種のプ
ログラミングシステムである。各生成対象毎に、各コー
ド生成のための規則をプログラム生成規則群6には、C言語
つ。即ち、図1に示すコード生成規則群6には、C言語
によるソースプログラムコード11を生成するためのC
言語ソースプログラムコード生成規則と、メカ制御専用
言語によるソースプログラムコード12を生成するための
メカ制御専用言語プログラムコード生成規則8と、シ

シミュレーション実行形式コード113を生成するためのシミュレーション実行形式コード生成規則9とを参照する構成になっている。各生成規則は、I F部とT H E N部とから構成されている。I F部にはマッピング条件のパターンリストを含める。T H E N部には当該条件にマッチした場合のソースプログラムコード等への変換方法を記述する。その具体的な内容は後で説明する。

[0026] 即ち、マトリクスコンパイラ4は、例えばマトリクスエディタ2により生成されたプログラム辞書をC言語によるソースプログラムコード11に変換する場合、C言語ソースプログラムコード生成規則7を参照しながら変換処理を進める。即ち、プログラム辞書の各項目と同一パターンとなるI F部を含む生成規則を、C言語ソースプログラムコード生成規則7から検索する。そして、該当する規則が見つかった場合、そのT H E N部のソースプログラムコードを出力として得る。こうして、プログラム辞書をC言語によるソースプログラムコード11に変換できる。メカ制御専用言語によるソースプログラムコード112の生成やシミュレーション実行形式コード113の生成も同様に行うことができる。

[0027] なお、以下の例では、ファームウェア全般にプログラム言語として汎用され、その中で最も普及しているC言語を例にして説明を行う。しかしながら、このような言語の如何を問わず、マトリクスコンパイラは対応するコード生成規則があれば、任意のソースプログラムコードを得ることが可能である。

[0028] (シミュレーションデバッガの機能) シミュレーションデバッガ14はマトリクスコンパイラ4が生成したシミュレーション実行形式コード113を解析し実行するインタプリタである。このようなシミュレーションデバッガ14は、従来よりプログラム開発支援に不可欠なものとして広く使用されてきている。その具体的な方法としては、プログラムを1ステップずつ実行させ結果を出力させたり、あるいはプログラム中にブレイクポイントを設定し、そこからプログラムを実行させ、ブレイクポイントでの出力や状態を表示させるといった用

本発明では、このようなシミュレーションを更に進め、メカ制御用ソフトウェアのリアルタイムなメカ動作のアニメーションを表示できる構成にしている。図1にシミュレーションデバッガの構造を示した。このシミュレーションデバッガと、状態表示部67を備えている。動作規則61やシミュレーション条件63等を元にメカ制御部66を生成する部分でメカ制御プログラム状態部69に表示する部分

状態に切り換える。MG2と指定したマグネットをONする。即ち、マグネットをONすることによって媒体をクランプして媒体の搬送を実行する。一方、センサ1がONでセンサ2がOFFの場合には、MG2と表示したマグネットをONし、次にそのマグネットをOFFする。また、センサ1も2もONの場合には、同様にMG2と表示したマグネットをONし、その次にこれをOFFする。

[0034] このように、搬送路上のマグネットのONとOFFを繰り返して、媒体をつかんで搬送路上に吸入する。次に、ステップS5において、搬送動作を行う。ここでは、媒体の先端をどこまで送り込むか、先端目標位置や媒体の後端目標位置、最大移動量等を規定した上で、搬送機構を起動させる。次のステップS6では、センサの状態が動作中の間、ステップS7に示すような手順が実行される。即ち、ここでは媒体先端位置をステップS7の状態で供給されるパルス数等でカウントし、その数が1227に達すると、記号MG3と表示したマグネットをONする。その他の場合は、MG3と表示したマグネットをOFFにしておく。

[0035] 以上のように、表形式によれば、非常に簡単に正確なプログラミングが可能となる。図13には、シミュレーションデバッガの出力例説明図を示す。上記のような媒体の搬送等のメカニズムを制御するプログラムでは、その動作を解析する場合、媒体の位置、搬送路の状態、センサ等の状態をプログラムの任意のステップで一見して理解できるように表示することが好ましい。そこで、本発明のシミュレーションデバッガは、図13に示すような搬送路70とその搬送路のスケール71とを表示する。1番に示したのは搬送路、2番目に示したのは搬送路のスケールで、搬送路のスケール71はステップシンボルモータに供給されるパルス数に基づいて表示されている。

[0036] また、搬送路70上には、媒体72の位置を抽出するセンサMS2、MS4、MS7等が設けられている。更に、媒体72を搬送駆動するローラを制御するためのマグネットMG2、MG3、MG4等が設けられている。これらの各マグネットがON/OFFする状態も、この図の下側に示すようにタイムチャートで表示される。タイムチャートのローレベルはOFF、ハイレベルはONの状態である。シミュレーションデバッガが各処理ステップ毎に、そのステップが終了したときの状態を図に示すような要領でアニメーション化する。そして、プログラマーの希望に従って表示する。これにより、媒体の挿入、検知、搬送、印字制御等を直接図解し、見ることで、プログラムの動作の効率化を図ることができる。

[0037] 図14には、図12のS1に示した部分からソースプログラムコード出力を得るための具体的なマトリクスコンパイラの動作説明図を図示した。図の

(a) はプログラム辞書、(b) は日本語変換辞書、(c) はマトリクスコンパイラのルール (シミュレーション実行形式コード生成規則)、(d) はソースプログラムコード出力を示している。図12のS1に示した命令は、「媒体状態=媒体無し」という命令である。これは、即ち「媒体状態」という変数に対し媒体無しというデータを入力するための命令である。このような表形式のデータは、図14(a)に示すように記述される。図の1行目のname \$whatは、この表の名称を表している。また、次の行のtype if-whatは、表の種類を示している。次のline with 168 topは、この表の幅を示している。

[0038] そして、この表の欄に記入された文字は5行目の"に挟まれた媒体状態=媒体無しという文字で表現されている。表の内容を記述する方法はどのような方法でもよいが、このような記述によって表のデータがコンピュータの認識できる文字データとなる。従って、逆にこのプログラム辞書を読み取るようにによって、図12に示すような表が描かれる。このようにプログラム辞書の作成方法自体は従来より表計算ソフトウェア等でもよく知られており、どのような方法を使用しても差し支えない。

[0039] ここで、このプログラム辞書からソースプログラムコードを得るために、(c)に示すマトリクスコンパイラのルールが適用される。ところが、ソースプログラムコードに日本語を含めようとした場合、日本語は使用できない場合がある。いずれの場合にも、日本語プログラムコードの生成を可能にするためには、プログラム辞書に含まれている日本語を英語等に交換しておくことが好ましい。このために、(b)に示す日本語変換辞書が使用される。従って、プログラム辞書の中で「媒体状態」という言葉と「媒体無し」という言葉が英語に変換される。

[0040] 日本語変換辞書はこのようにするためのために、表の中で使用される言葉に対応する英語を用意する。図の矢印によって示した「媒体状態」という単語は、b_statusという英語に置き換えられる。「媒体無し」という単語はこの日本語変換辞書には用意されていない。そこで、マトリクスコンパイラは自動的にこの媒体無しという言葉に対応する変数、他の言葉を重複しないように設定しようとする。例えば、その変数をJg67と定める。このような準備を行った後、(c)に示すルールが適用される。マトリクスコンパイラには、多数のルールが設けられており、先に説明したように、その1F部が適用される場合に、T H E N部に置き換えられたソースプログラムコードが出力される。

[0041] この(c)の1行目は表の名前で、(a)に示した2行目と一致する。1F部には%keyという言葉がある。これは1つの命令であって、その次に続くbodyという言葉を見つけたらという命令である。

11

同様にして、thenという言葉とwhatという言葉を続けて見つけなさいという命令が続く。そして、whatという言葉を見つけた場合に、これに続く言葉を&restという変数に代入しなさいという命令になる。これによって&restという変数に「媒体状態=媒体無し」という部分が代入される。なお、これは、先に説明したように、先にb_status=j g 67という式に変換されている。

【0042】ここで、(c)に示すTHEN部が実行される。即ち、ここでは、appendという命令によって&restという変数の中身に“.”を後ろに付けて出力するといった内容になる。これは、即ち、(d)に示すようなソースプログラムコードとして出力することになる。これによって、図14(a)に示すようなソースプログラム辞書からこの(d)に示すようなソースプログラムコードが得られる。他の部分についても全く同様プログラム辞書に含まれた日本語を一旦英語に変換し、その後マトリクスコンパイラに用意された各種のルールに従って各部を順次変換することによってソースプログラムコードが得られる。

【0043】図15は日本語変換辞書の一部の例を示し、図16～図20は図12の表に対応するその他の部分のプログラム辞書の例を示す。また、図21、図22は、図12のS4の部分を変換するために使用されるルールのリストであり、図23、図24は、マトリクスコンパイラの出力として得られたソースプログラムコードの例を示す。プログラム辞書中、S4と示した部分は、図21、図22に示すマトリクスコンパイラのルールに従って変換され、図23、図24に示すS4と印をしたソースプログラムコードに置き換えられる。その手順等は先に説明した通りである。

【0044】本発明は以上の実施例に限定されない。上記実施例においては、メカ動作制御のためのプログラム作成に本発明を利用した例を示したが、その他、例えばワードプロセッサのような文書データの加工処理、その他各種の情報処理のためのプログラム作成に利用できる。また、プログラム言語はC言語を用いて説明したが、C言語の改良型であるC++、その他C言語系のプログラムやこれ以外の各種の汎用されているプログラム作成に利用することが可能である。

【0045】

【発明の効果】以上説明した本発明のプログラム開発支援システムによれば、プログラムのロジックの表記を表形式により分かり易く行い、これを自動的にプログラムソースコードに変換する構成にしたので、プログラムの表現について専門的な知識を持たないものにも分かり易い扱いの無いプログラムの記述ができる。また、第三者も容易に分かり易いプログラムの記述が可能となる。更に、プログラムの各命令が実行されたときに、制御対象装置の各部の状態を表す状態情報等を具体的に表示する

12

シミュレーションデバッグを用意することによって、プログラムの開発をより容易に具体的に行的に行うことが可能になる。

【図面の簡単な説明】

【図1】本発明のプログラム開発支援システム実施例を示すブロック図である。

【図2】本発明の実施のためのハードウェア説明図である。

【図3】モジュール定義図の説明図である。

【図4】データ形式定義図の説明図である。

【図5】構造体定義図の説明図である。

【図6】データ意味定義図の説明図である。

【図7】表の説明図である。

【図8】表の基本形説明図(その1)である。

【図9】表の基本形説明図(その2)である。

【図10】シミュレーションデバッグの構造を示すブロック図である。

【図11】具体的な操作画面例(その1)である。

【図12】具体的な操作画面例(その2)である。

【図13】シミュレーションデバッグの出力例説明図である。

【図14】具体的なマトリクスコンパイラの動作説明図である。

【図15】日本語変換辞書の例説明図である。

【図16】プログラム辞書の例説明図(その1)である。

【図17】プログラム辞書の例説明図(その2)である。

【図18】プログラム辞書の例説明図(その3)である。

【図19】プログラム辞書の例説明図(その4)である。

【図20】プログラム辞書の例説明図(その5)である。

【図21】マトリクスコンパイラのルールの例説明図(その1)である。

【図22】マトリクスコンパイラのルールの例説明図(その2)である。

【図23】ソースプログラムコード出力の例説明図(その1)である。

【図24】ソースプログラムコード出力の例説明図(その2)である。

【符号の説明】

1 コンソール

2 マトリクスエディタ

3 プログラム辞書

4 マトリクスコンパイラ

6 コード生成規則群

11 C言語によるソースプログラムコード

12 メカ制御専用言語によるソースプログラムコード

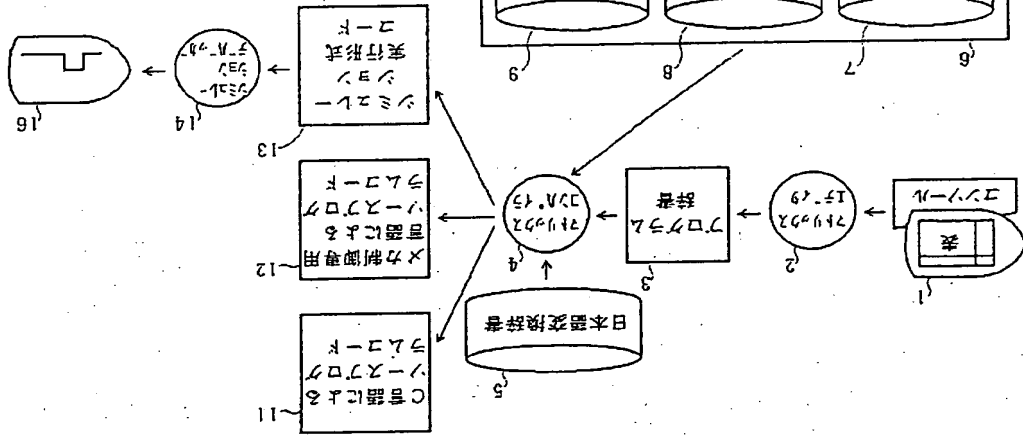
13

13 シミュレーション実行形式コード

14

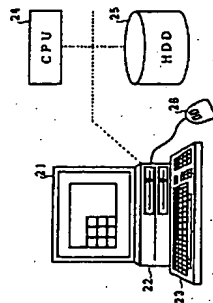
14 シミュレーションデバッグ

【図1】



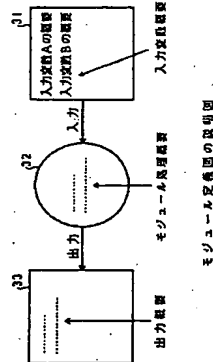
本発明のプログラム開発支援システムブロック図

【図2】



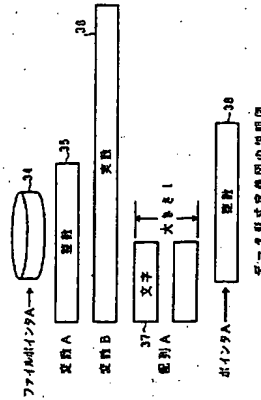
本発明の実施のためのハードウェア説明図

【図3】



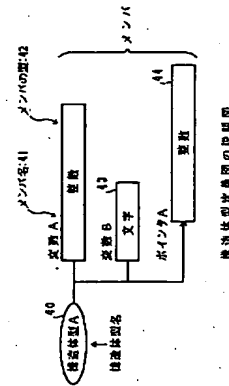
モジュール記述図の説明図

【図4】



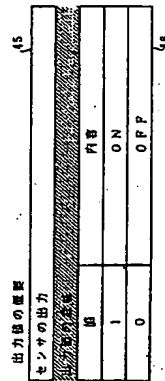
データ形式変換図の説明図

【図5】



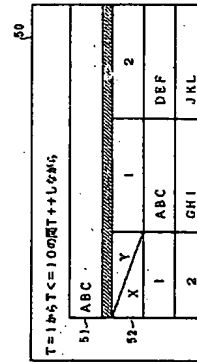
モジュール記述図の説明図

【図6】



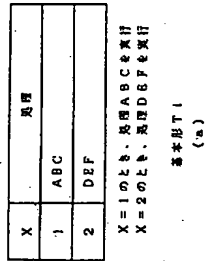
データ形式変換図の説明図

【図7】



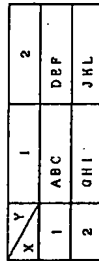
表の表記図

【図8】



X=1のとき、処理ABCを実行
X=2のとき、処理DEFを実行

基本形T1
(a)

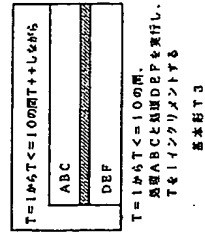


X=1, Y=1のとき、処理ABCを実行
X=1, Y=2のとき、処理DEFを実行
X=2, Y=1のとき、処理GHIを実行
X=2, Y=2のとき、処理JKLを実行

基本形T2
(b)

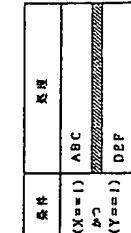
表の表記図(その1)

【図9】



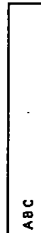
T=1からT<=10の間T+1しながら
処理ABCと処理DEFを実行し、
Tをインクリメントする

基本形T3
(a)



X=1かつY=2のとき、
処理ABCと処理DEFを実行

基本形T4
(b)

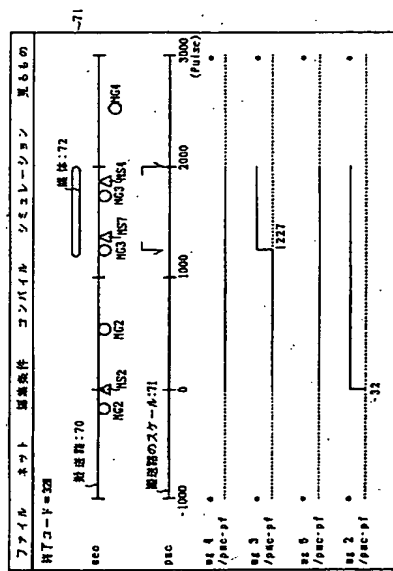


無条件に処理ABCを実行

基本形T5
(c)

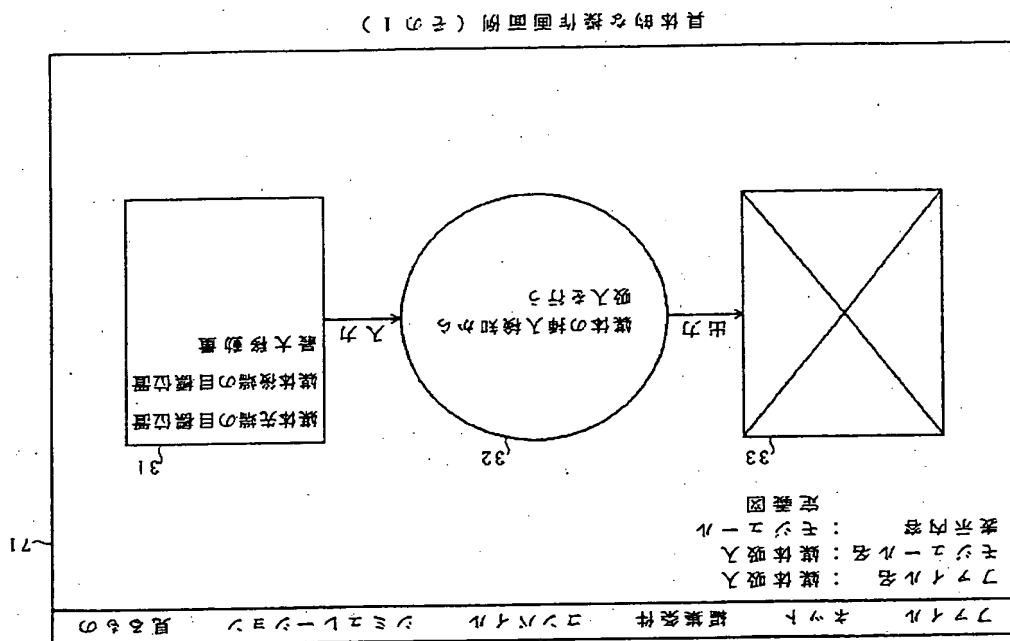
表の表記図(その2)

【図13】

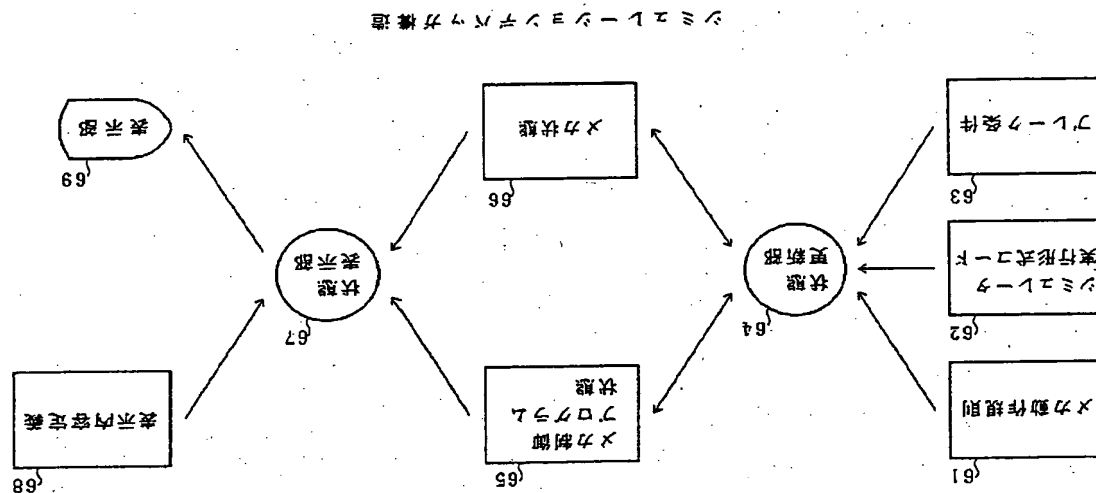


シミュレーションプログラムの出力例
△: センサ O: マグネット

【図11】



【図10】



【図15】

(整数)	"int"
(文字)	"char"
(実数)	"float"
(符号なし整数)	"unsigned int"
(符号なし文字)	"unsigned char"
(符号なし short)	"unsigned short"
(倍長実数)	"double"
(メイン)	"main"
(構造体)	"struct"
(リターン)	"return"
(ブレーク)	"break"
(自動変数)	nil
(外部変数)	"extern"
(静的変数)	"static"
(レジスタ変数)	"register"
(その他)	"default"
(改行)	"printf"
(表示)	"Can't open file"
(エラー)	"EVENT"
(状態)	"STATUS"
(オン)	"1"
(オフ)	"0"
(先端位置)	"front_pos"
(センサ 1)	"sensor_1"
(センサ 2)	"sensor_2"
(センサ)	"sensor"
(マトリックス)	"matrix"
(クランプ)	"brump"
(排出)	"haiyutu"
(吸入終了位置)	"kyunyu_end_pos"
(停止)	"stop"
(1/16パルス吸入)	"kyunyu_1pis"
(1/16パルス排出)	"haiyutu_1pis"
(媒体状態)	"b_status"
(媒体先端位置)	"p"
(媒体後端位置)	"pt"
(先端位置リード)	"read_pos"
(動作中)	"01"
(モータ状態)	"motor_status"
(センサーセンス)	"sensor_sence"
(ビット 0)	"0x0001"

→ これが数明に対応する項目です。

日本語変換辞書の例説明図

【図16】

全体	=((prog_name "プログラム名1") (define ("マクロ定義" (macro -----))) (var -----) (struct -----) (module -----)))	プログラム名 ... マクロ定義 ... グローバル変数定義 ... 構造体定義 ... モジュール定義
module	=((name "モジュール名1") (type "...") (abstract "...") (input -----) (output -----) (var -----) (struct -----) (table -----)))	モジュール名 ... モジュールの型 ... モジュールの概要 ... 入力変数定義 ... 出力値定義 ... ローカル変数定義 ... 構造体定義 ... 表の定義
struct	=((name "構造体名1") (member ((name "メンバ名1") (type "...") (set -----) (pointer -) (length "...") (struct-name "...") (lval -----)) ((name "メンバ名2") (type "...") (set -----) (pointer -) (length "...") (struct-name "...") (lval -----))))	構造体名 ... メンバ名 ... メンバの型 ... ポインタか？ ... 配列時の大きさ ... 構造体型名 ... メンバの値の意味
var	=((name "変数1") (type "...") (set "...") (pointer -) (length "...") (struct-name "...") (lval -----))	変数名 ... 変数の型 ... 記憶クラス ... ポインタか？ ... 配列時の大きさ ... 構造体型名 ... 変数の値の意味
table	=((name "表の名前1") (type "...") (l-width 表の幅 何列目) (body -----)))	表の名前 ... 表の形式 ... 表示情報 ... 表形式にあった情報

プロログラム読書例 (その1)

プログラム辞書例 (その1)

【図 24】

```

b_status = Jg87;
while (b_status==Jg87){
  Jg85();
  switch (sensor_1){
    case OFF:
      switch (sensor_2){
        case OFF:
          ;
          break;
        case ON:
          b_status=crusp;
          Jg86(WG2);
          break;
      }
    case ON:
      break;
      switch(sensor_2){
        case OFF:
          Jg86(WG2);
          Jg87(WG2);
          break;
        case ON:
          Jg86(WG2);
          Jg87(WG2);
          break;
      }
      break;
  }
}
Jg88(Jg72, Jg77, Jg81);
while(motor_status=01)
  if (1227<pf)
    Jg86(WG3);
  else
    Jg87(WG3);
}

```

S4

ソースプログラムコード例（その2）